Data Blocks

To save data storage space, you can create a global data area with data blocks.

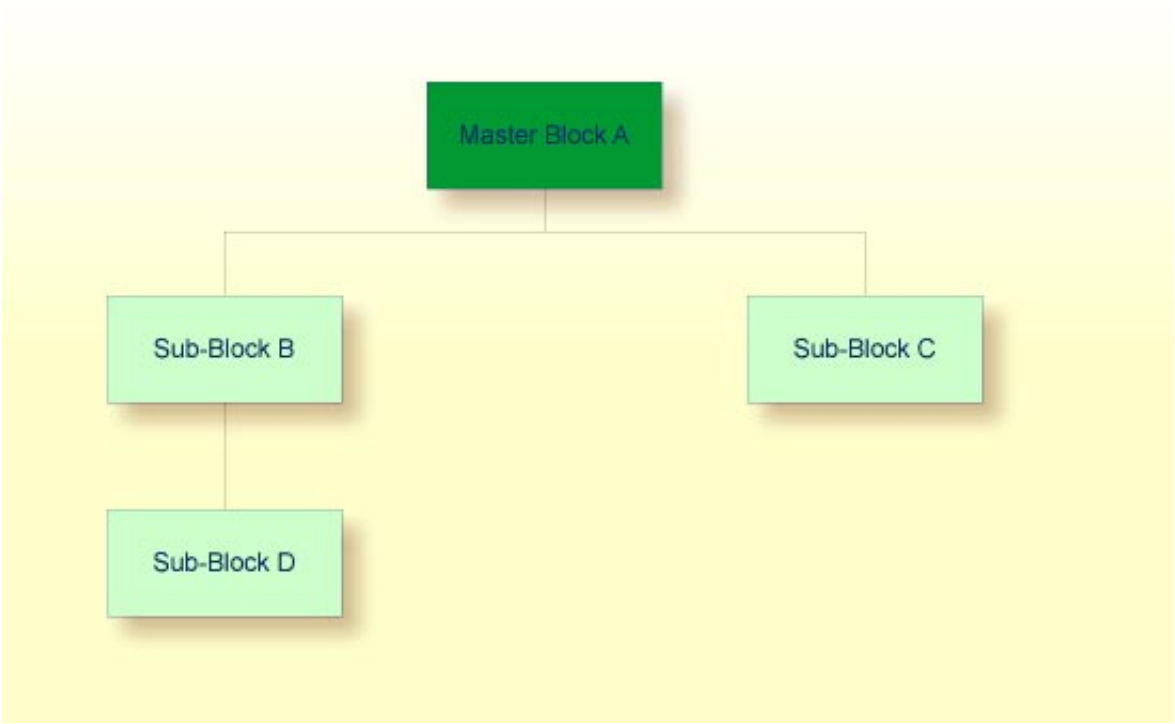
The following topics are covered:

- Example of Data Block Usage
- Defining Data Blocks
- Block Hierarchies

Example of Data Block Usage

Data blocks can overlay each other during program execution, thereby saving storage space.

For example, given the following hierarchy, Blocks B and C would be assigned the same storage area. Thus it would not be possible for Blocks B and C to be in use at the same time. Modifying Block B would result in destroying the contents of Block C.



Defining Data Blocks

You define data blocks in the data area editor. You establish the block hierarchy by specifying which block is subordinate to which: you do this by entering the name of the "parent" block in the comment field of the block definition.

In the following example, SUB-BLOCKB and SUB-BLOCKC are subordinate to MASTER-BLOCKA; SUB-BLOCKD is subordinate to SUB-BLOCKB.

The maximum number of block levels is 8 (including the master block).

Example:

Global Data Area G-BLOCK:

```
I T L Name                            F Leng Index/Init/EM/Name/Comment
- - - -------------------------------- - ---- --------------------------------
  B   MASTER-BLOCKA
    1 MB-DATA01                       A   10
  B   SUB-BLOCKB                                MASTER-BLOCKA
    1 SBB-DATA01                      A   20
  B   SUB-BLOCKC                                MASTER-BLOCKA
    1 SBC-DATA01                      A   40
  B   SUB-BLOCKD                                SUB-BLOCKB
    1 SBD-DATA01                      A   40
```

To make the specific blocks available to a program, you use the following syntax in the DEFINE DATA statement:

Program 1:

```
DEFINE DATA GLOBAL
       USING G-BLOCK
       WITH MASTER-BLOCKA
END-DEFINE
```

Program 2:

```
DEFINE DATA GLOBAL
       USING G-BLOCK
       WITH MASTER-BLOCKA.SUB-BLOCKB
END-DEFINE
```

Program 3:

```
DEFINE DATA GLOBAL
       USING G-BLOCK
       WITH MASTER-BLOCKA.SUB-BLOCKC
END-DEFINE
```

Program 4:

```
DEFINE DATA GLOBAL
       USING G-BLOCK
       WITH MASTER-BLOCKA.SUB-BLOCKB.SUB-BLOCKD
END-DEFINE
```

With this structure, Program 1 can share the data in MASTER-BLOCKA with Program 2, Program 3 or Program 4. However, Programs 2 and 3 cannot share the data areas of SUB-BLOCKB and SUB-BLOCKC because these data blocks are defined at the same level of the structure and thus occupy the same storage area.

Block Hierarchies

Care needs to be taken when using data block hierarchies. Let us assume the following scenario with three programs using a data block hierarchy:

Program 1:


```
DEFINE DATA GLOBAL
       USING G-BLOCK
       WITH MASTER-BLOCKA.SUB-BLOCKB
END-DEFINE
*
MOVE 1234 TO SBB-DATA01
FETCH 'PROGRAM2'
END
```

Program 2:

```
DEFINE DATA GLOBAL
       USING G-BLOCK
       WITH MASTER-BLOCKA
END-DEFINE
*
FETCH 'PROGRAM3'
END
```

Program 3:

```
DEFINE DATA GLOBAL
       USING G-BLOCK
       WITH MASTER-BLOCKA.SUB-BLOCKB
END-DEFINE
*
WRITE SBB-DATA01
END
```

Explanation

- Program 1 uses the global data area G-BLOCK with MASTER-BLOCKA and SUB-BLOCKB. The program modifies a field in SUB-BLOCKB and FETCHes Program 2 which specifies only MASTER-BLOCKA in its data definition.
- Program 2 resets (deletes the contents of) SUB-BLOCKB. The reason is that a program on Level 1 (for example, a program called with a FETCH statement) resets any data blocks that are subordinate to the blocks it defines in its own data definition.
- Program 2 now FETCHes program 3 which is to display the field modified in Program 1, but it returns an empty screen.

For details on program levels, see Multiple Levels of Invoked Objects.

